WHAT IS CLAIMED IS:

1. A method of interfacing a front-end systems layer with a back-end systems layer using a self describing data structure, the method comprising:

a) receiving a request from a front-end systems layer;

b) translating the request;

c) executing custom application code to access data within a back-end systems layer as a function of the translated request;

d) receiving data in response to the translated request from the custom application code; and

e) translating the data to a format defined in the request.

2. The method of claim 1, wherein a) comprises generating the request in a servlet request format.

3 The method of claim 1, wherein b) comprises translating the request to extensible markup language.

4. The method of claim 1, wherein b) comprises translating the request into a document object model document to represent an input message.

5. The method of claim 1, wherein b) comprises limiting the translated request to representation as at least one of integer, long, Boolean, string and group fields.

6. The method of claim 1, wherein b) comprises generating a plurality of fields as a function of tags provided in the request.

7. The method of claim 1, wherein b) comprises selectively setting the length of a field name for each of a plurality of fields.

8. The method of claim 1, wherein d) comprises establishing a structure for the response in extensible mark up language.

9. The method of claim 1, wherein d) comprises limiting the data to representation as at least one of integer, long, Boolean, string and group.

10. The method of claim 1, wherein d) comprises generating a document object model document to represent an output message as a function of data received.

11. The method of claim 1, wherein d) comprises translating to one of hypertext markup language, extensible markup language and website meta language.

12. The method of claim 1, further comprising f) returning the translated data to the front-end systems layer.

13. A method of leveraging extensible markup language technology to interface a front-end systems layer with a back-end systems layer, the method comprising:

a) receiving a request initiated with a delivery technology;

b) identifying the value of a request name parameter from the request;

c) translating the request to an input message, the input message comprising a root element and a plurality of sub elements; and

d) initiating the retrieval of data as a function of the request name parameter.

14. The method of claim 13 further comprising:

e) providing data as a response;

f) creating an output message with the response, the output message comprising a root element and a plurality of sub-elements; and

g) translating the output message to a format compatible with the delivery technology.

15. The method of claim 13, wherein c) comprises setting the root element to a message name as a function of the request name parameter.

16. The method of claim 13, wherein c) comprises creating a document object model document.

17. The method of claim 13, wherein d) comprises executing custom application code corresponding to the request name parameter.

18. The method of claim 14, wherein f) comprises creating a document object model document.

19. The method of claim 14, wherein f) comprises setting the root element to a message name as a function of the request name parameter.

20. The method of claim 13, wherein the delivery technology comprises at least one of an Internet browser, a telephone, a wireline communication device, a wireless communication device and a wireless application protocol device.

21. A method of operating a business services application for retrieving data with delivery technologies, the method comprising:

a) developing custom application code in a subclass of a BusinessService class, the custom application code responsive to a request for data initiated by the delivery technologies;

b) translating the request to a first document object model document with an ApiService class;

c) selectively limiting the data structure of the first document object model document as a function of a Message class and a Field class;

d) executing the custom application code to retrieve data as a function of the first document object model document;

e) reading data into a second document object model document with the ApiService class;

f) selectively limiting the data structure of the second document object model document as a function of the Message class and the Field class; and